Pinctrl and GPIO - Interactions and Footguns ELCE 2024 Vienna

Chen-Yu Tsai, ChromeOS

Disclaimer

- Opinions are my own

Agenda

- Motivation
- GPIO and Pinctrl Subsystems
- The Hardware
- Subsystem and Userspace Interactions
- Incorrect Driver Implementations and Usage
- Suggestions

Motivation for this Presentation

- Pinmux `strict` mode vs GPIO function pin mux in DT
  - Most platforms prefer the latter
- Incorrect / conflicting pin config debugfs readbacks on some platforms
- Attempts to understand what the input and output pin config options mean
  - Long thread asking the maintainer
  - https://lore.kernel.org/all/CAGb2v66XpjvDnTpi2=SPbeAPf844FLCai6YFwvVqvmF9z4Mj=A@mail.gmail.com/
- Userspace GPIO footgun

GPIO subsystem

- Provides common interface to control GPIO pins
- Drive line level as output
- Read line level as input
- IRQ on input level or change
- Control or emulate open drain/source modes
- Configure *basic* biasing (pull-up / pull-down)
- Ensure exclusivity of GPIO usage
    - Non-exclusive GPIO usage is only used for fixed regulators

Pinctrl subsystem

- Pinmux
  - Configure each pin for a specific function or peripheral interface
  - Maintain exclusive usage for each pin
- Pin Config
  - Configure electrical characteristics of each pin
    - `PIN_CONFIG_BIAS_*` - Bias (pull-up / pull-down) or lack thereof
    - `PIN_CONFIG_DRIVE_STRENGTH*` - Drive strength
    - `PIN_CONFIG_INPUT_ENABLE` - Input enable
    - `PIN_CONFIG_OUTPUT_ENABLE` - Output enable
      - This is only loosely defined
    - `PIN_CONFIG_OUTPUT` - Output enable with level
    - Etc.

GPIO (or PIO) Hardware

- General Purpose I/O or Programmable I/O pins
- Hardware designers sometimes refer to any pin that can be configured through software as a GPIO pin
- Implements the GPIO and Pin Control functions
- GPIO input/output and line value control may be either a muxed in function or directly attached to the line
    - If directly attached, could be a special mode or always active

Pin Control and GPIO Interactions

- Interactions between the two subsystems are loosely defined
  - To support all different hardware designs
  - Depending on device tree and driver usage ordering, may leave hardware in an inconsistent state
- Pinmux and GPIO exclusivity are separately controlled
  - Unless `.strict` field in `struct pinmux_ops` is set

GPIO stepping over Pinctrl

- Prerequisite: GPIO is a "muxed in" pin function
- Driver A requests a set of pins muxed for its function
- Driver B (or userspace) requests GPIO for one of those pins, overriding the muxing, and breaking device A
  - Commonly seen when GPIO or pin numbers are incorrectly entered into the device tree
  - Common workaround is to also request the pin mux in addition to the GPIO
    - Doesn't help the userspace request case, which has no pin mux request tied to it

Pin Config and Pin Mux

Remember `PIN_CONFIG_OUTPUT` and `PIN_CONFIG_OUTPUT_ENABLE` from earlier?

- Implemented for Rockchip platforms as "Mux to GPIO, set direction to output (, and set value)"
  - Pin config settings are applied after pin mux
    - Overrides the original pin mux setting
    - But only ever seen used with GPIO pin muxes, making the muxing here redundant?
- Implemented for some other platforms as "set direction to output (and set value)"
  - Does nothing if pin isn't muxed to GPIO function
  - Can lead to bogus configurations

Output Enable == "GPIO as Output"

- Many platforms implement `OUTPUT_ENABLE` as "set GPIO to output"
  - MediaTek (fixed!)
  - Meson (Amlogic)
  - Qualcomm SoC
    - Also implements `INPUT_ENABLE` as "disable output" and is even documented!
  - Qualcomm PMIC
    - Also maps GPIO operations to pin config settings
- Implement `INPUT_ENABLE` as inverse of `OUTPUT_ENABLE`
  - Microsemi Ocelot pin controller
    - Ignores pin config argument
  - Cy8c95x0 GPIO expander
  - Awinic AW9523 I2C GPIO expander

Pin Config Readback

- Pin Config state can be read back from debugfs
- Some drivers implement read back differently from configuring

  `pinctrl: mediatek: paris: Fix PIN_CONFIG_INPUT_SCHMITT_ENABLE readback`
  `pinctrl: mediatek: paris: Rework support for PIN_CONFIG_{INPUT,OUTPUT}_ENABLE`
- Readback may report bogus state
  - Reports GPIO direction or value when pin is not muxed as GPIO

Usage with No Effect (MediaTek)

```
uart1-pins-default {
        pins-rx {
                pinmux = <PINMUX_GPIO121__FUNC_URXD1>;
                input-enable;
                bias-pull-up;
        };
        pins-rts {
                pinmux = <PINMUX_GPIO47__FUNC_URTS1>;
                output-enable;
        };
        ...
};
```

From `arch/arm64/boot/dts/mediatek/mt8183-kukui.dtsi` before it was fixed

Suggestions

- Implement *strict* mode in drivers
  - Possibly made compatible with DTs that also specify pinmuxes for GPIO usage
  - Also provides pin mux lock outs against userspace usage
- Implement `PIN_CONFIG_OUTPUT*` correctly without conflicts with GPIO or pin muxing
  - Or remove them from the drivers

Questions?

Thank You